Mixed-instance querying: a lightweight integration architecture for data journalism

Raphaël Bonaque, Tien Duc Cao, Bogdan Cautis, François Goasdoué, Javier Letelier, Ioana Manolescu, Oscar Mendoza, Swen Ribeiro, Xavier Tannier, Michaël Thomazo

Mixed-instance querying: a lightweight integration architecture for data journalism

R. Bonaque[1,2] T. D. Cao[1,2] B. Cautis[2,3,4] F. Goasdoué[5] J. Letelier[1,2]
I. Manolescu[1,2] O. Mendoza[1,2] S. Ribeiro[2,4,6] X. Tannier[2,4,6] M. Thomazo[1,2]

[1]INRIA & LIX, École Polytechnique [2]U. Paris-Saclay [3]LRI [4]U. Paris-Sud [5]IRISA, U. Rennes 1 & INRIA [6]LIMSI, CNRS

ABSTRACT

As the world's affairs get increasingly more digital, timely production and consumption of news require to efficiently and quickly exploit heterogeneous data sources. Discussions with journalists revealed that content management tools currently at their disposal fall very short of expectations. We demonstrate TATOOINE, a lightweight data integration prototype, which allows to quickly set up integration queries across (very) heterogeneous data sources, capitalizing on the many data links (joins) available in this application domain. Our demonstration is based on scenarios we study in collaboration with Le Monde, France's major newspaper.

1. MOTIVATION AND OUTLINE

Data, social networks, and the media As the world's affairs get increasingly digital, the production and consumption of news are massively impacted. First, *data that journalists may work on is more and more digitized*: *structured* databases, such as government-gathered data (demographics, economics, taxes, elections, etc.), legal records, stock quotes for companies; *un-structured* and *semi-structured* datasets, all can be exploited for news analysis. For instance, *large-scale RDF datasets, endowed with ontologies*, such as DBPedia or Yago, incorporate significant amounts of manual and automatic effort that led to structuring information about a wide variety of places, people, major events, and concepts. These sources of open data can be readily queried through SPARQL endpoints, e.g., data sources in the Linked Open Data cloud (lod-cloud.net). High-quality content may also be organized in *structured text* form (HTML or XML), e.g., (interrelated) laws and regulations, public speeches etc. Second, *the arena of interactions between organizations* (countries, government, corporations, media) *and/or individuals* (from politicians to bloggers), which journalism follows and analyzes,*is increasingly digital*, with the Web and social networks as communication means of choice. The latter allows journalists to examine connections between public and unknown figures in social networks[1], and analyze online interaction traces between these actors.

The need for dynamic data journalism tools A novel generation of data journalists has emerged, well aware of the potential of computerized databases, and eager to combine in innovative ways both static and dynamic information coming from structured, semi-structured, and un-structured databases and social feeds. Nevertheless, feedback collected through discussions with the data journalism and fact-checking team Les Décodeurs (lemonde.fr/les-decodeurs) of the major French journal "Le Monde", as well as with journalists from "The Washington Post" and "The Financial Times" reveal that current content management tools at their disposal are disparate and limited, falling very short of expectations. Furthermore, the very short news cycles for many news stories, the frequent emergence of new sources, as well as the eagerness to be the first to exploit them, do not entice journalists nor media companies to invest time, effort, and funding into devising and filling a "standard" data warehouse comprising all types of information.

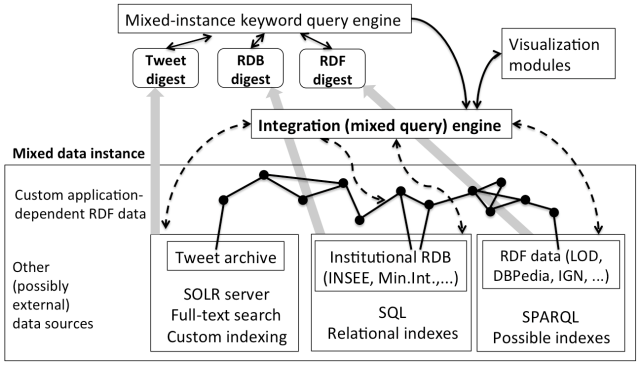

Figure 1: Mixed instance and querying in TATOOINE.

Our approach: lightweight integration architecture We propose to demonstrate TATOOINE, a lightweight data integration prototype we have devised based on our discussions with Les Decodeurs. TATOOINE allows journalists to exploit heterogeneous data sources of different data models, which we view as a *mixed data instance*. Within the instance, we distinguish (Figure 1) a set of (structured, un-structured, or semi-structured) *data sources*, of various data models, each of which resides within a system providing some query capabilities over its data. Examples in the figure are: a set of Tweets stored within Apache Solr, a full-text search platform; a set of relational curated databases, such as those provided by INSEE, the French institute of economic and social statistics, or the Ministry of Interior, which compiles detailed results of national and regional elections; and a set of RDF data sources, such as French territory description data from the National Geographic Institute (IGN), and LOD sources, in particular DBPedia, etc.

The mixed instance also comprises a *custom (application-dependent) RDF data graph* depicted under the form of black round nodes connected by edges above the data sources. This graph comprises an ontology (set of classes and properties) and/or data triples specific to a certain data journalism domain. For instance, when working on an article about French politics, a journalist may load a specialized data source classifying French political parties into currents and giving the affiliation of each party in the European Parliament; as there are very few parties, such a data source is typically put together manually. Another data source may hold all the elected representatives of the people, as they can be extracted from

[1]A Twitter account has been correctly identified as belonging to French politician Marine LePen, by analyzing its posts and its social connections: bit.ly/1Lxi27C.

websites such as nosdeputes.fr and nossenateurs.fr; we found out Les Décodeurs had scraped these as a one-time effort, and stored them in a simple tabular file. Whatever the sector, be it Middle East politics, EU education, or French agriculture, journalists do tend to have small-to-significant size data sources, typically under the form of a text or tabular file, which can be easily exported into RDF.

A crucial remark we made while looking at the journalists' most commonly used data sources is that *repeated values across sources are frequent*. This is due to several reasons, among which: (i) *link-richness*: a public figure's profile on, say, Twitter, also provides his Facebook ID, his web page, and possibly the web page of his main political party. This is because public figures want to attract attention, be easy to find, and easy to associate (in the reader's mind) to their organizations, such as parties or NGOs; (ii) *common naming for humans*: media authors, whether public figures, communication managers, journalists, etc. have a strong interest in allowing readers to easily relate an article to a topic. Thus, staple terms such as "unemployment" or "migrants" are consistently used by publishers hoping to attract readers; the usage of Twitter hashtags is another striking example; (iii) *common naming for machines* (part of the so-called Search Engine Optimizations), where publishers pick non-ambiguous, name-rich titles and representative article snippets to facilitate indexing and lookup through search engines. For instance, "Sarkozy visits Libya" is systematically preferred to the more ambiguous "The President visits Country at War". Common naming for machines also includes *codes* (such as "MSFT" for Microsoft in NASDAQ, or "75" for the French department enclosing Paris) defined by the administration or corporations, which are widely adopted and frequently used in databases and media.

RDF data often features *repeated Uniform Resource Identifiers (URIs)*, since RDF advocates (re)using URIs to refer to resources. Moreover, *RDF graphs may also include non-URI values*, such as names or codes, appearing in other data sources. In Fig. 1, joins between the custom RDF graph and the sources are depicted by RDF edges having one end "in a data source". When the custom graph joins with several data sources, it effectively provides a "bridge" across them. RDF is a model of choice for expressing such bridges, as its flexibility (lack of structural constraints) makes it easy to add to a graph any piece (no matter how small) of useful information.

Mixed querying in TATOOINE To exploit the join opportunities provided by these repeated values (URIs, names, keywords, codes), we devise our *integration (mixed query) engine* (Figure 1) as a mediator, running on top of the different data sources, oftentimes (but not necessarily) interconnected by means of custom application-dependent RDF data. The engine evaluates *mixed queries*, which combine sub-queries expressed in the query language(s) of several heterogeneous sources, and RDF querying on the custom data. A sample query is, for instance: *"for a given hashtag and each political affiliation (left, right etc.), find the most prolific tweet authors of that affiliation having used that hashtag, and their Facebook accounts"*. Mixed queries can be used also to query dynamically discovered datasets, e.g., the address of a relational database is found in an INSEE table and part of the mixed query is shipped there for evaluation. Writing such queries requires some database skills[2]. To further simplify the interaction with the mixed data instance, we have also devised a *keyword-based query engine*, which exploits *data source digests* TATOOINE computes from the sources. Based on these digests, the keyword-based query engine identifies a set of mixed queries which, evaluated over the set of (joining) datasets, return the results users are interested in. Finally, data visualizations can be devised to interact with (the results of queries over) mixed instances; these are particularly effective data journalism tools.

Our contributions are: (i) an RDF-centric lightweight integration approach for an arbitrarily varied set of data sources (with heterogeneous models and query languages which we exploit as much as possible) and (ii) a novel approach for keyword search on such a heterogeneous instance, based on source digests, making it easy for non-expert users to discover valuable connections in the data.

[2]Some data journalists are very tech-savvy, but many others clearly prefer just *using* (and imagining!) data management tools.

2. MIXED CONTENT QUERYING

In this section, we introduce now mixed content instances (Section 2.1) and tools for querying them (Section 2.2).

2.1 Mixed content

RDF data and queries RDF is a W3C standard to describe Web data and knowledge. It allows expressing *triples*, of the form $\mathrm{s\ p\ o}$, meaning that subject s has the property p whose value is o. Sample data description using RDF triples are $\mathrm{LeMonde\ foundedIn\ "1944"}$ and $\mathrm{Samuel\ worksFor\ LeMonde}$, where plain strings designate URIs, e.g., Samuel and foundedIn, while quoted strings denote constants (a.k.a. literals). The RDF standard also provides a set of *built-in URIs* to further enrich data descriptions; in particular, $\mathrm{rdf{:}type}$ allows assigning types, called *classes*, to resources as in $\mathrm{Samuel\ rdf{:}type\ Journalist}$. RDF Schema can express *domain specific knowledge about the classes and properties* used in data descriptions notably using four popular properties: $\mathrm{rdfs{:}subclass}$ and $\mathrm{rdfs{:}subproperty}$ state that a class or property is a specialization (particular case) of another; $\mathrm{rdfs{:}domain}$ and $\mathrm{rdfs{:}range}$ type respectively the subjects or objects of a property. E.g., *journalists are employees* translates to $\mathrm{Journalist\ rdfs{:}subclass\ Employee}$, *one works for whom pays him* to $\mathrm{worksFor\ rdfs{:}subproperty\ paidBy}$, *only organizations have founding dates* to $\mathrm{foundedIn\ rdfs{:}domain\ Organization}$, and *people only work for organizations* to $\mathrm{worksFor\ rdfs{:}range\ Organization}$.

Importantly, given a set of triples, called an *RDF graph* (graph, in short), knowledge triples may beget *implicit* triples, which do hold in the graph though not explicitly present. For example, within a graph comprising the above triples, $\mathrm{Samuel\ paidBy\ LeMonde}$, $\mathrm{Samuel\ rdf{:}type\ Employee}$, $\mathrm{LeMonde\ rdf{:}type\ Organization}$, etc. hold. *RDF entailment* rules, part of the RDF standard, show how to derive the (finite) unique set of implicit triples of a graph.

For a given RDF graph G, we denote by G^∞ its *saturation* (or *sematics*) , i.e., the result of adding to G all derivable implicit triples.

A prominent language for querying RDF graphs is that of *basic graph pattern queries* (BGPs, in short), the SPARQL subset corresponding to conjunctive queries). A BGP is of the form:

$$q(\bar{x})\text{:-}\ t_1, \ldots, t_n$$

where $q(\bar{x})$ is the query head, with q the query name and $\bar{x}$ its output variables projected from the query body $t_1, \ldots, t_n$, which is a conjunction of *triple patterns*, i.e., triples whose subject, property and object positions can also hold variables.

The *evaluation* of a BGP q on a graph G is classically defined as the *set* of $\phi(\bar{x})$ tuples obtained by all possible embeddings ϕ of q into G. The *answer* of q over G is defined as the evaluation q against G^∞, i.e., all the triples, explicit or implicit, holding in G.

SPARQL queries also allow disjunction, optional pattern matching, construction of RDF triple results, aggregation etc.

Mixed instances We assume a given set of *data models*, and one or several *query languages* for each. For example, an RDF source may accept SPARQL, an XML one may accept XPath or XQuery.

DEFINITION 2.1 (MIXED INSTANCE). *A mixed instance $I = (G, D)$ is an RDF graph G together with a set of data sources D.*

```
{ "created_at":"Tue March 01 03:42:31 +0000 2016",
  "id":464244242167342513,
  "text":"Je suis là aujourd'hui pour montrer qu'il y a
 une solidarité nationale. En défendant ... #SIA2016",
  "user":{
    "id":483794260,
    "name":"François Hollande",
    "screen_name":"fhollande",
    "description":"Président de la République française",
    "followers_count":1502835},
  "retweet_count":469, "favorite_count":883,
  "entities":{"hashtags":["SIA2016",],"urls":[]} }
```


Figure 2: Sample tweet from the mixed data instance I.

As a running example, we consider a mixed instance I as sketched in Figure 1: its RDF graph G contains information on entities related to the French political landscape such as

POL01140 rdf:type politician
POL01140 position headOfState
POL01140 foaf:name "François Hollande"
POL01140 twitterAccount "fhollande"

G also contains metadata on other RDF or relational data sources from I, such as the INSEE data source d_{INSEE} containing for instance an SQL table for the *Production and value-added of the agriculture in 2015* (bit.ly/1QqcIlx). Finally, I contains a Solr database of tweets, exemplified (in JSON) in Figure 2.

2.2 Mixed instance querying

To query mixed instances, we first propose *Conjunctive Mixed Queries* (CMQ, in short) of the form:

$$q(\bar{x}) \text{:-} \; q_G(\bar{x_0}), q_1(\bar{x_1})[d_1], \ldots, q_n(\bar{x_n})[d_n]$$

with, similar to BGPs, $q(\bar{x})$ being the query head, q the query name, and $\bar{x}$ its output variables projected from the body. However, the body is a conjunction of a query q_G over the custom application dependent RDF graph G of a mixed instance $I = (G, D)$ (recall Fig. 1) and, optionally, of $q_1, \ldots, q_n$ queries over D sources. Importantly, each of $d_1, \ldots, d_n$ is either a source URI or a q variable (i.e., in $\bigcup_{i=0}^{n} \bar{x}_i$). CMQs generalize BGPs against RDF graphs.

The *answers* to q are the *set* of $\phi(\bar{x})$ tuples obtained through all possible embeddings ϕ of q into I, i.e., for which (i) $\phi(\bar{x_0})$ is an answer to q_G against G *and* (ii) for every i, $\phi(\bar{x_i})$ is an answer to q_i against d_i if d_i is a source URI, or against $\phi(d_i)$ if d_i a variable. Observe that this allows to identify at runtime, for each embedding ϕ, a potentially different data source d_i.

A sample CMQ looking for tweets from head of states about "SIA2016" (*Salon International de L'Agriculture*) is:

$$q_{\text{SIA}}(t, id) \text{:-} \; q_G(id), tweetContains(t, id, SIA2016)[d_{\text{Solr}}]$$

where $q_G(id)$:- x position headOfState, x twitterAccount id and the Solr query $tweetContains$ yields pairs (t, id) such that the tweet t with the hashtag "SIA2016" was posted on the account id. If d_{Solr} is a URI, then $tweetContains$ is only evaluated on the source identified by this URI, otherwise it is evaluated on every data source of the mixed instance that accepts it.

Keyword-based querying Mixed queries are powerful and flexible, but writing them requires query language skills and knowledge of the custom RDF graph. This may be hard for journalists, thus we devise an approach to query over mixed instances through *keyword queries*. From each source, we build a *digest*, comprising: (i) its *schema* (if it has one; otherwise we use data-derived structural summaries, i.e., XML or JSON Dataguides, RDF summaries [3] etc.) and (ii) a *representation of the set of atomic values* (constants, URIs etc.) associated to each position in the schema. This could mean: the values of the attribute a in a relation R; the values found in the `user.screenname` attributes of all the tweets; the values of property position etc.. If value set representations are provided by the data sources (e.g., an index on $R.a$), TATOOINE exploits them; if not, TATOOINE builds and stores them externally. The precision level of the value set representations is controlled by parameters dividing up the available space; histograms and Bloom filters are used. We view all digests as directed graphs (e.g., for a relational database, there is one node per attribute, one edge per key-foreign key constraint, etc.), and to each node we attach the representation of the set of data values corresponding to it.

Given the search keywords "head of state" and "SIA2016", the engine looks up the keywords in (value set representations from) the digests; in our example, "head of state" appears in the custom RDF RDF part, while "SIA2016" appears as a hashtag. Following the approach of [9], the shortest join paths from "head of state" to "SIA2016" are then identified, which may be through the id of a head of state, then his Twitter id (these are found in the custom RDF), to the tweet tagged "SIA2016" as hashtag (found in Solr). Thus, the structured query q_{SIA} is generated from the keywords.

2.3 Evaluating mixed instance queries

A mixed query is decomposed into a query over the custom RDF and one query per data source. Subqueries are then ordered for evaluation such that (i) bindings for data sources must be obtained before the source can be queried, (ii) parallelism is exploited when possible, and (iii) the most selective subqueries are executed first, in classical mediator style, and extending our experience with mixed XML-RDF querying [7]. The remaining processing (joins etc.) on subquery results takes place within our in-house iterator-based execution engine (Java, approx. 10K lines).

3. DEMONSTRATION SCENARIO

Dataset We will use a mixed instance we built over the last year as part of our collaboration with Les Décodeurs. It comprises: (i) tweets from about 4,500 French politicians and political groups, updated daily since June 2015 (1.6M tweets at the time of writing). Author, retweet number, timestamp, and stemmed text are continuously indexed by an Apache Solr engine; (ii) 10K Facebook posts collected from verified pages of the most important French politicians since June 2015. Author, creation, and collection timestamp, stemmed text, number of likes, shares, and comments associated with each post are also indexed by a Solr instance; (iii) an RDF graph containing basic (name, gender, date and place of birth, nationality, spouse, etc.) and detailed (DBPedia URI, personal website, Twitter ID, Facebook ID, current political position, party affiliations, parliament and senate group affiliations etc.) information of top French politicians, as well as political parties and currents.

Scenarios demonstrated through mixed queries prepared before the presentation include:

(1) Identify factual sources of information that relate to the claims made by a personality on Twitter, for instance the French President.

(2) Compare vocabulary used by different parties on a user-defined topic, and show the most influential tweets on this topic (illustrated by tag clouds). All terms w used by each party P in a set of tweets Q (result of a mixed query) are ranked by their exponentiated *pointwise mutual information* (PMI, in short), comparing the probability of w in the party to its global probability in the entire corpus. Probabilities are calculated with Maximum Likelihood Estimation [10]:

$$PMI(w, Q) = \frac{\sum_{t \in P} n_{tw}}{\sum_{t \in P} n_t} \frac{N_Q}{n_{Qw}}$$

where t is a tweet from party P, n_{tw} is the count of word w in tweet t, n_t the number of words in tweet t, N_Q the total number of words in Q and n_{Qw} the count of word w in Q.

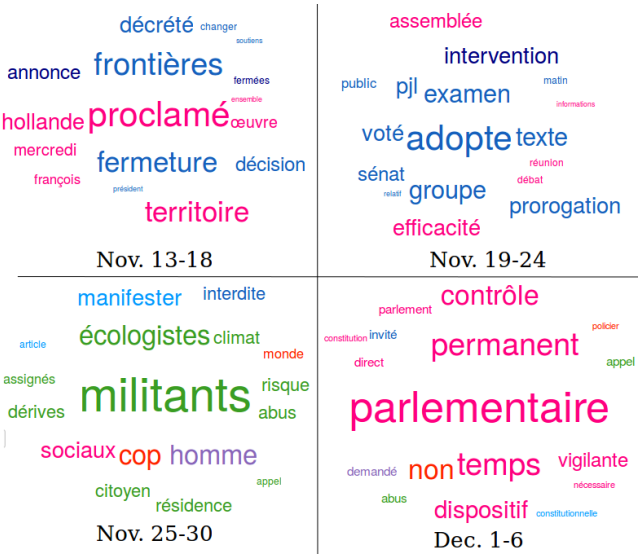

Figure 3: Tag cloud visualization of mixed query results on the tweet database.

Figure 3 provides a screenshot of the current tag cloud visualization tool. It shows the *weekly evolution of French politician vocabulary* on the state of emergency decreed in France after the November 2015 attacks, colored according to the political group of the author: extreme-left in red, left in pink, right in blue, extreme-right in dark blue, ecologists in green. This visualization highlights the evolution of the public discourse in time. The first week is factual (tweets reflect the events); the second is institutional (the parliament votes the state of emergency); objections (*abuses, excesses, risk*) raised by ecologists appear in the third; in the fourth, the left parties ask for *vigilance* and *control* etc.

The audience will also be prompted to create new queries by customizing existing ones, and/or through keywords, and see the digests matches, before triggering the execution.

4. RELATED WORK

TATOOINE enables querying a diverse set of data sources, close in spirit to global-as-view data integration [5]. A particularity of TATOOINE is the *preeminence given to the RDF model* as "glue" between all others; however, in contrast to previous integration systems exporting all data sources as semistructured graphs [4], TATOOINE is based on the juxtaposition of different-model queries integrated through joins on common variables; unlike this and other previous systems, it also allows dynamic source discovery. We are interested in RDF as it simplifies the creation of integration links between datasets, by its built-in concept of identifiers (URIs) and its *linked* aspect. RDF ontologies are also handy means to integrate different databases under a common conceptual view; if TATOOINE is queried through an ontology, the setting resembles the well-known ontology-based data access (OBDA). Clearly, the general data integration setting raises many other issues. *Mappings* can be used to relate data sources to each other or to an integration schema, or to migrate data between sources [1, 8]; entity resolution is also an important area. TATOOINE does not currently use mappings nor an integrated global schema, since it appeared more urgent to provide the journalists with query tools capable of combining several sources in order to get results, and with a simpler keyword-based query interface. This fits their data usage patterns, consisting first of searching (for specific items and for data interconnection), and then (if the search is successful) of refining the respective queries and/or generating visualisations based on them.

Recent works seek to efficiently exploit heterogeneous-model data sources ; under the names of polystores or hybrid stores [11, 6] assume sources are (possibly trivially mapped to) relations, while [2] supports heterogeneous sources through complex view-based rewriting. In contrast, TATOOINE natively supports a variety of heterogeneous models and tools, inspired from the data journalists' needs, and exploits the opportunities offered by RDF for bridging across data sources. Work on TATOOINE continues within our collaborative French R&D project with Les Décodeurs (also supported by a Google Research Award on Computational Journalism).

5. ACKNOWLEDGMENTS

This work was partially supported by the projects ANR-11-LABEX-0045-DIGICOSME (part of ANR-11-IDEX-0003-02), ANR-15-CE23-0025, ANR-PIA Datalyse, ANR-13-CORD-0020 ALICIA, a Google Research Award in Computational Journalism.

6. REFERENCES

[1] Z. Bellahsene, A. Bonifati, and E. Rahm, editors. *Schema Matching and Mapping*. Springer, 2011.

[2] F. Bugiotti, D. Bursztyn, A. Deutsch, I. Ileana, and I. Manolescu. Invisible Glue: Scalable Self-Tuning Multi-Stores. In *CIDR (online proceedings)*, 2015.

[3] S. Cebirić, F. Goasdoué, and I. Manolescu. Query-oriented summarization of RDF graphs. *Proc. VLDB Endow.*, 8(12):2012–2015, Aug. 2015.

[4] S. S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. D. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *IPSJ*, pages 7–18, 1994.

[5] A. Doan, A. Y. Halevy, and Z. G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.

[6] A. Elmore, J. Duggan, M. Stonebraker, and al. A demonstration of the BigDAWG polystore system. *Proc. VLDB Endow.*, 8(12):1908–1911, Aug. 2015.

[7] F. Goasdoué, K. Karanasos, Y. Katsis, J. Leblay, I. Manolescu, and S. Zampetakis. Growing triples on trees: An XML-RDF hybrid model for annotated documents. *The VLDB Journal*, 22(5):589–613, Oct. 2013.

[8] L. M. Haas, M. A. Hernández, H. Ho, L. Popa, and M. Roth. Clio grows up: From research prototype to industrial tool. In *SIGMOD*, pages 805–810, 2005.

[9] W. Le, F. Li, A. Kementsietsidis, and S. Duan. Scalable keyword search on large RDF data. *IEEE Trans. Knowl. Data Eng.*, 26(11):2774–2788, 2014.

[10] L. Le Cam. "Maximum likelihood — an introduction". *ISI Review*, 58:153–170, 1990.

[11] J. LeFevre, J. Sankaranarayanan, H. Hacigumus, and al. MISO: Souping up big data query processing with a multistore system. In *SIGMOD*, pages 1591–1602, 2014.